CSE 428, Section 1, Spring 2002. Final exam, 29 April 2002

This exam consists of 5 questions, totaling 100 points. Write your answers directly on these pages. **Please remember to write your name and student ID at the top of each sheet. If you are in Section 2 (morning section) then cross out the phrase "Section 1" and write "Section 2" instead.** This exam is *closed book*.

Question 1, Java (20 points)

For each of the following sentences about Java, say whether it is true or false.

1. If a class has several synchronized methods, there will be several locks associated to the class, one for each method.

2. A lock can be obtained only by one thread at a time (and so only one thread at the time can execute a synchronized method on a given object).

3. A command of the form `t.wait()`, where `t` is a thread, will suspend `t` (assuming that the method `wait()` is the standard one, i.e. it has not been redefined).

4. When a thread suspends, it releases all the locks that it was holding.

5. When a thread is resumed, its execution will be restarted immediately (i.e. it will be immediately scheduled for execution).

6. An event handler does not need to be defined as a thread.

7. In event programming, the program in general does not specify when an event occurs, but only how to react when the event occurs.

8. In event programming, a typical event would be the change of state of an internal variable.

9. An exception is called "checked" if the compiler checks that whenever there is the possibility that the exception will be raised, then the exception will be caught somewhere.

10. An exception like "NumberFormatException", when it is raised, will automatically send the control back to the instruction that has raised the exception, and that instruction will be executed again.

Question 2, Memory management (20 points)

Consider the following C++ program, which contains the definition of a class queue:

```
class Queue {
private:
   class Element {
   public:
      int info;
      Element* next;
      Element(int k, Element* n) { info = k; next = n; }
   };

   Element* front;
   Element* rear;

public:
   Queue() { front = rear = NULL; }
   ~Queue() { while (front!=NULL) pop(); }
   void insert(int k) {
      if (rear==NULL)
         rear = front = new Element(k,NULL);
      else
         rear = rear->next = new Element(k,NULL);
   }
   int pop() { // the case of an empty Queue is not considered here,
               // but you should consider it in your definition of Queue
      Element* temp = front;
      int n = front->info;
      front = front->next;
      if (front==NULL) rear = NULL;
      delete temp;
      return n;
   }
};

void insertnumbers(Queue* q, int n){
   if (n==0)
      cout<<''Checkpoint 1\n'';
   else {
      q->insert(n);
      insertnumbers(q, n-1);
   }
};

void main() {
   Queue* q = new Queue();
   insertnumbers(q,3);
   cout << ``Checkpoint 2\n'';
   delete q;
   cout << ``Checkpoint 3\n'';
}
```

Illustrate with three drawings the situations in memory (stack and heap) at the moment in which the instruction `cout << "Checkpoint 1 \n";`, `cout << "Checkpoint 2 \n";`, and `cout << "Checkpoint 3 \n";` are executed, respectively. Please make explicit all the informations stored in the activation records and in the (allocated part of the) heap at that moment. Use arrows for pointers. Remember that in C++ there is no static link. You can also omit the control link and the temporaries.

Please state explicitly whether there are memory leaks or dangling pointers.

Question 3, Basic ML programming (20 points)

Consider the following datatype for binary trees that contain integers at their leafs.

```
datatype tree = Leaf of int | Node of tree * tree;
```

Write in ML the definition of the following three functions.

1. The function `maxtree` of type `tree -> int` which returns the maximum integer value of all integers in the leafs of the tree given as input.

2. The function `sumtree` of type `tree -> int` which returns the sum of all the integers in the tree given as input.

3. The function `counttree` of type `tree -> tree` which maps a tree to another tree in which the leafs are numbered from left to right (beginning with 1). The value of the integers in the input tree are ignored.

For example, the following values should be computed with these functions.

```
- val ex1 = Node(Node(Node(Leaf  5, Leaf 6), Node(Leaf 10, Leaf 1)), Leaf 8);
val ex1 = Node (Node (Node (Leaf 5,Leaf 6),Node (Leaf 10,Leaf 1)),Leaf 8) : tree
- maxtree ex1;
val it = 10 : int
- sumtree ex1;
val it = 30 : int
- counttree ex1;
val it = Node (Node (Node (Leaf 1,Leaf 2),Node (Leaf 3,Leaf 4)),Leaf 5) : tree
```

Question 4, Types and higher-order programming in ML (20 points)

Part A: Write in SML code to define the function `composelist` which takes a list of functions and composes them, as follows. The value of

$$\texttt{composelist}\ [f_1, f_2, \ldots, f_n]\ x$$

should equal $f_1(f_2(\ldots(f_n(x))\ldots))$. If $n = 0$, then the result of composing should be the identify function; that is, this displayed value should be simply x. **Also:** after writing down the function's definition, indicate what type ML will infer for this function. (Notice that the function is written in Curried syntax.)

Part B: The following code declares one datatype and three functions.

```
datatype numb = I of int | R of real;

fun cases f g (I x) = f x
  | cases f g (R x) = g x;

fun diag x = [I x, R x];

fun plus(I x, I y) = I(x+y)
  | plus(I x, R y) = R(real x + y)
  | plus(R x, R y) = R(x+y);
```

Write down the type for the three defined functions that is inferred by ML. If there is a type error, then indicate that. If there are any warning printed, then indicate that as well.

1. The type for `cases`: (list also any error or warning messages)

2. The type for `diag`: (list also any error or warning messages)

3. The type for `plus`: (list also any error or warning messages)

Question 5, Basic Prolog programming (20 points)

Write code for this question in Prolog. If you need any auxiliary predicates that are not built-in (eg, comparison predicates such as `<` and `>` and `=` or the `is` predicate), then you will need to define those predicates.

1. Write code to specify the predicate `firstlast(List,First,Last)` so that this holds if and only if `List` is a list and `First` is the first element of that list and `Last` is the last element of that list.

2. Write code to specify the predicate `maxlist(List,Max)` so that this holds if and only if `List` is a list of non-negative integers and `Max` is the largest member of `List`. You may assume that `List` is a list of non-negative integers and that the max of the empty list can be taken as 0.

3. Write code to specify the predicate `selectalternate(List,AList)` so that this hold if and only if `List` is a list and `AList` is the list built from the first, third, fifth, etc, members of `List`. In other words, `AList` results from deleting from `List` those items in even numbered positions in `List`.